Lab 3: Packet Capture (IDS)

Details

Aim: To provide define the usage of an intrusion detection system

Activities

1. The WinPcap library can be used to read the source and destination IP addresses and TCP ports. For this the **TCPPacket** class is used. Initially modify the program in Lab 2 so that it now displays the source and destination IP and TCP ports [1]:

http://www.dcs.napier.ac.uk/~bill/WinPCap3.zip

```
private static void device_PcapOnPacketArrival(object sender, Packet packet)
{
   if(packet is TCPPacket)
   {
      DateTime time = packet.PcapHeader.Date;
      int len = packet.PcapHeader.PacketLength;

      TCPPacket tcp = (TCPPacket)packet;
      string srcIp = tcp.SourceAddress;
      string dstIp = tcp.DestinationAddress;
      int srcPort = tcp.SourcePort;
      int dstPort = tcp.DestinationPort;

      Console.WriteLine("{0}:{1} -> {2}:{3}", srcIp, srcPort, dstIp, dstPort);
   }
}
```

A sample run, using a Web browser connected to google.com gives:

```
84.53.143.151:80 -> 192.168.1.101:3582
84.53.143.151:80 -> 192.168.1.101:3582
192.168.1.101:3582 -> 84.53.143.151:80
```

Where it can be seen that the WWW server TCP port is 80, and the local port is 3582. Run the program, and generate some network activity, and determine the output.

Determine the output of the test run:

2. Modify the program in 3.12.1, so that it only displays traffic which is *distended* for a Web server. Prove its operation.

How was the code modified:

3. Next modify the code so that it detects only ICMP packets (using the **ICMPPacket** class), and displays the source and the destination addresses, along with the TTL (time-to-live) value [1]:

```
private static void device_PcapOnPacketArrival(object sender, Packet packet)
{
   if(packet is ICMPPacket)
   {
      DateTime time = packet.PcapHeader.Date;
      int len = packet.PcapHeader.PacketLength;

      ICMPPacket icmp = (ICMPPacket)packet;
      string srcIp=icmp.DestinationAddress;
      string dstIp=icmp.SourceAddress;
      string ttl=icmp.TimeToLive.ToString();

      Console.WriteLine("{0}->{1} TTL:{2}", srcIp, dstIp, ttl);
   }
}
```

A sample run is shown next for a ping on node 192.168.1.102:

```
Press any <RETURN> to exit
192.168.1.101->192.168.1.102 TTL:128
192.168.1.102->192.168.1.101 TTL:128
192.168.1.101->192.168.1.102 TTL:128
```

Run the program, and ping a node on the network. What is the output, and why does it show three responses for every ping:

4. Modify the program in 3.12.2, so that it displays the Ethernet details of the data frame, such as [4]:

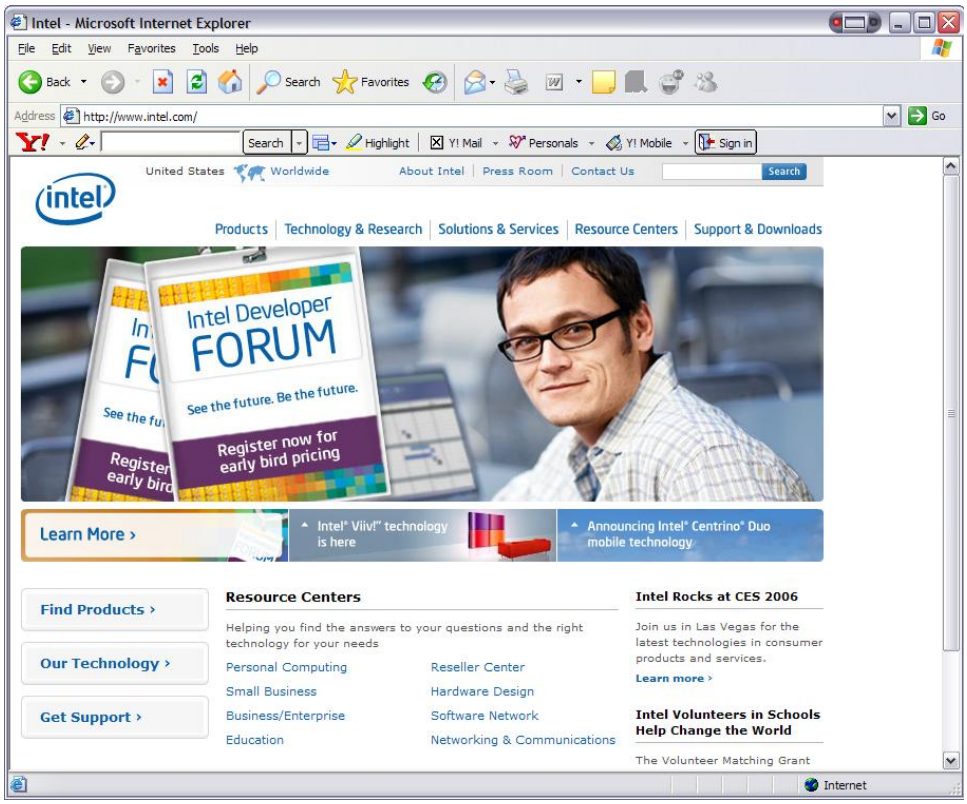
```
private static void device_PcapOnPacketArrival(object sender, Packet packet)
{
   if( packet is EthernetPacket )
   {
      EthernetPacket etherFrame = (EthernetPacket)packet;
      Console.WriteLine("At: {0}:{1}: MAC:{2} -> MAC:{3}",
         etherFrame.PcapHeader.Date.ToString(),
         etherFrame.PcapHeader.Date.Millisecond,
         etherFrame.SourceHwAddress,
         etherFrame.DestinationHwAddress);
   }
}
```

5. It is possible to read the contents of the data package by converting it to a byte array (using the Data property), and then convert it to a string, such as:

```
private static void device_PcapOnPacketArrival(object sender, Packet packet)
{
   if(packet is TCPPacket)
```

```
    {
       DateTime time = packet.PcapHeader.Date;
       int len = packet.PcapHeader.PacketLength;

       TCPPacket tcp = (TCPPacket)packet;

       byte [] b = tcp.Data;

       System.Text.ASCIIEncoding format = new System.Text.ASCIIEncoding();

       string s = format.GetString(b);

       s=s.ToLower();

       if (s.IndexOf("intel")>0) Console.WriteLine("Intel found...");
    }
}
```

The above code detects the presence of the word Intel in the data packet. Run the program, and then load a site with the word Intel in it, and prove that it works, such as for:




```
Intel found...
Intel found...
```

Did the code work:

6. It is then possible to filter for source and destination ports, and with source and destination addresses. For example, the following detects the word Intel on the destination port of 80:

```
private static void device_PcapOnPacketArrival(object sender, Packet packet)
{
   if (packet is TCPPacket)
   {
      DateTime time = packet.PcapHeader.Date;
      int len = packet.PcapHeader.PacketLength;

      TCPPacket tcp = (TCPPacket)packet;
      int destPort = tcp.SourcePort;

      byte [] b = tcp.Data;

      System.Text.ASCIIEncoding format = new System.Text.ASCIIEncoding();

      string s = format.GetString(b);

      s=s.ToLower();

      if (destPort==80 && (s.IndexOf("intel")>0))
         Console.WriteLine("Intel found in outgoing on port 80...");
   }
}
```

Did the code work:

7. A key indication of network traffic is in the TCP flags. The following determines when the SYN flag is detected, and also the SYN, ACK flags:

```
if(packet is TCPPacket)
{
   DateTime time = packet.PcapHeader.Date;
   int len = packet.PcapHeader.PacketLength;

   TCPPacket tcp = (TCPPacket)packet;
   int destPort = tcp.SourcePort;

   if (tcp.Syn) Console.WriteLine("SYN request");
   if (tcp.Syn && tcp.Ack) Console.WriteLine("SYN and ACK");

}
```

Prove the operation of the code, and modify it so that it detects a SYN request to a Web server (port: 80), and displays the destination IP address of the Web server.

Outline the code used:

8. Modify the code in 7 so that it displays all the flags for data packets.

[1] This code is based on the code wrapper for WinPCap developed by T.Gal [http://www.thecodeproject.com/csharp/sharppcap.asp].